42392.P9769 *PATENT*

UNITED STATES DESIGN PATENT APPLICATION

FOR

MECHANISM FOR PROVIDING POWER MANAGEMENT THROUGH VIRTUALIZATION

Inventors:

MICHAEL A. KOZUCH

STEPHEN T. CHOU

ERIK COTA-ROBLES

STALINSELVARAJ JEYASINGH

ALAIN KAGI

GILBERT NEIGER

SEBASTIAN SCHOENBERG

RICHARD UHLIG

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard, Seventh Floor
Los Angeles, CA 90025-1026

(408) 720-8300

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number EL672750861US
Date of Deposit December 27, 2000
I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Michelle Begay
(Typed or printed name of person mailing paper or fee)

Michelle Begay
(Signature of person mailing paper or fee)

MECHANISM FOR PROVIDING POWER MANAGEMENT THROUGH VIRTUALIZATION

Field of the Invention

The present invention relates generally to virtual machines, and more specifically to providing power management via a virtual machine monitor.

Background of the Invention

A conventional virtual machine monitor (VMM) runs on a computer, hereafter called the "host platform", and presents to other software the abstraction of one or more virtual machines (VMs). Each VM functions as a self-contained computer, running its own "guest operating system" (guest OS), which can be a standard OS for the computer being virtualized (e.g., Microsoft® Windows® for a Personal Computer). Currently, each guest OS is responsible for solving power management problems. However, some OSes are unable to manage the power consumed by the host platform because they are not equipped to handle power-management signals sent by host platform hardware. In addition, the guest OS expects to run on a dedicated computer rather than in a VM and is unaware of other VMs that may be running on the same host platform. As a result the guest OS may, in its attempts to provide power management of the VM that it is running in, conflict with the power management actions or expectations of other guest OSes running in other VMs. Accordingly, guest OSes running in multiple VMs cannot be allowed to directly provide power management of host platform hardware resources.

Therefore, there is a need for an alternative power management mechanism that will provide more efficient use of computing resources in a virtual machine environment.

Brief Description of the Drawings

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 is a block diagram of a system for providing power management via virtualization, according to one embodiment of the present invention;

Figure 2 is a flow diagram of a method for providing power management, according to one embodiment of the present invention;

Figure 3 is a flow diagram of a method for reducing resource requirements of virtual machines, according to one embodiment of the present invention;

Figure 4 is a flow diagram of a method for assisting a guest operating system, according to one embodiment of the present invention; and

Figure 5 is a block diagram of one embodiment of a processing system.

Description of Embodiments

A method and apparatus for providing power management via virtualization are described. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention can be practiced without these specific details.

Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that

throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, may refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus. Instructions are executable using one or more processing devices (e.g., processors, central processing units, etc.).

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose machines may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized

apparatus to perform the required method steps. The required structure for a variety of these machines will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

In the following detailed description of the embodiments, reference is made to the accompanying drawings that show, by way of illustration, specific embodiments in which the invention may be practiced. In the drawings, like numerals describe substantially similar components throughout the several views. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention. Other embodiments may be utilized and structural, logical, and electrical changes may be made without departing from the scope of the present invention. Moreover, it is to be understood that the various embodiments of the invention, although different, are not necessarily mutually exclusive. For example, a particular feature, structure, or characteristic described in one embodiment may be included within other embodiments. The following detailed description is, therefore, not to be taken in a limiting sense, and the scope of the present invention is defined only by the appended claims, along with the full scope of equivalents to which such claims are entitled.

The method and apparatus of the present invention provide a power management mechanism that can be used in a virtual machine environment. **Figure 1** illustrates a system 100 for providing power management via

virtualization, according to one embodiment of the present invention. In this embodiment, host platform 108 is a computing platform that comprises electronic hardware. In one embodiment, the electronic hardware consists of one or more power-manageable devices. These power-manageable devices may include, for example, a disk drive, a processor, or any other device capable of operating in a mode other than ON and OFF modes. For instance, a disk drive may be commanded to enter a sleep state in which the platters stop rotating, or a processor may be commanded to enter one of several power-consumption modes. The power consumption of the processor may be reduced by simultaneously reducing the voltage and frequency supplied, although other mechanisms for reducing the power consumption of the processor may be employed. In another embodiment, the host platform 108 also includes non-power-manageable devices. Such devices can typically operate either in ON mode or OFF mode and do not have a reduced power-consumption state.

The host platform 108 is capable of executing a virtual machine monitor (VMM) 104. The VMM 104, though typically implemented in software, exports a bare machine interface to higher level software. The interface exported by VMM 104 to the multiple VMs 102 may mirror the actual platform, so that it is virtualized, or it may differ in some or all respects so that a different platform is emulated. Such higher level software may comprise a standard or real-time OS, although the invention is not limited in scope in this respect and, alternatively, for example, the VMM 104 may be run within, or on top of, another VMM. VMMs and their typical features and

functionality are well-known by those skilled in the art and may be implemented, for example, in software, firmware or by a combination of various techniques.

As described above, the VMM 104 presents to other software (i.e., "guest" software) the abstraction of one or more virtual machines (VMs). **Figure 1** shows multiple VMs 102. Each VM 102 runs its own guest operating system (guest OS). In one embodiment, all guest OSes are capable of handling power-management signals sent by the host platform 108. In another embodiment, none of the guest OSes in the system 100 is capable of handling such signals. For instance, if the guest OS was developed before power-management features were added to a particular device or class of devices, said guest OS would not typically be able to adjust the power setting of that device. In yet another embodiment, one or more VMs 102 run guest OSs that have the capacity of handling the power-manageable signals and the remaining VMs 102 run guest OSs that lack such capacity.

The guest OS is provided with the illusion of executing on the host platform, rather than in a virtual platform. In one embodiment, the virtual abstraction presented to the guest OS matches the characteristics of the host platform 108. Alternatively, the virtual abstraction presented to the guest OS differs from the characteristics of the host platform 108.

The VMM 104 provides protection between VMs 102 and observes the activities of the VMs 102. In one embodiment, the VMM 104 includes a resource watch module 106 which monitors utilization of host platform devices by the VMs 102 and provides input pertaining to the allocation of the

host resources to the VMM 104. Based on this input, the VMM 104 manages power consumption of physical devices within the host platform 108. For instance, the resource watch module 106 may determine that a particular power-manageable device is not being utilized and provide this information to the VMM 104. The VMM 104 may then place this device in a reduced power-consumption state.

In one embodiment, the resource watch module 106 observes utilization of host platform devices whenever any VM 102 is started or stopped. In one embodiment, the resource watch module 106 determines that the VM 102 executes an application that does not require a certain device (e.g., a display device). The resource watch module 106 notifies the VMM 104 about this determination. The VMM 104 then provides to the corresponding guest OS only the abstraction of the remaining host platform devices rather than the entire host platform 108. In another embodiment, if the resource watch module 106 determines that the resources of a particular platform device have not been allocated to any of the VMs 102, the VMM 104 commands this device to enter a reduced power-consumption mode.

In an alternative embodiment, the resource watch module 106 constantly monitors the utilization of the host platform devices by the VMs 102. This embodiment is referred to herein as a dynamic power management of the host platform 108. **Figure 2** is a flow diagram of a method 200 for providing dynamic power management, according to one embodiment of the present invention.

Referring to **Figure 2**, method 200 begins with monitoring requests for computing resources of a host platform. The resource requests are initiated by one or more VMs. At processing block 206, the utilization of a host platform device is determined using the requests of one or more VMs for computing resources. In one embodiment, the utilization of the device by the VMs is determined by identifying a change in the operation of one or more VMs and deciding whether the change in the operation will affect the utilization of the device.

At decision box 208, a determination is made as to whether the device is fully utilized. If the determination is positive, i.e., the VMs fully utilize the capacity of the device, then the power-consumption state of the device remains unchanged. Otherwise, the power-consumption state of the device is modified (processing block 210). For instance, if the determination is made that the device is under-utilized, the device is placed in a reduced power-consumption state. Alternatively, the power-consumption state of the device may be modified to allocate more resources of the device to the VMs.

In one embodiment, the VMM notifies one or more VMs that support such a notification about the modification of the power-consumption state of the device. For those VMs that do not support the notification, the guest OSs may experience longer latencies when accessing the device that has been placed in a reduced power-consumption state.

Figure 3 is a flow diagram of a method 300 for reducing resource requirements of VMs, according to one embodiment of the present invention. In this embodiment, if the power available to the host platform decreases,

method 300 allows reducing the resource requirements of VMs by stopping one or more VMs that are not being used.

Referring to **Figure 3**, method 300 begins with identifying a decrease in the power available to the host platform (processing block 304). For instance, the decrease may occur because the host platform is reduced from AC power and is now running on battery power. At decision box 306, a determination is made as to whether any of the VMs is quiescent. If the determination is positive, i.e., a quiescent VM is found, the VMM then saves the current state information of this VM (processing block 308) and stops this VM (processing block 310). As a result, the resources allocated to the VM are freed. The VMM may then reduce the power consumption of one or more devices which were partially or fully allocated to the saved VM, thereby adjusting to the decrease in the available power. If more than one quiescent VMs are found, the VMM may save and stop as many of these quiescent VMs as necessary to avoid exceeding the power available to the host platform. Then, at decision box 312, a determination is made as to whether the VMs that remain active still exceed the power available to the host platform. If the determination is negative, method 300 stops. Otherwise, method 300 proceeds to processing block 314.

If either the determination made at decision box 306 is negative (i.e., none of the VMs is quiescent) or the determination made at decision box 312 is positive (i.e., after one or more quiescent VMs have been stopped, the remaining VMs still exceed the power available to the host platform), then the active VMs are examined. In particular, at processing box 314, the VMM

determines which subsets of the active VMs can remain active without exceeding the power available to the host platform. For instance, the VMM may make this determination by evaluating all possible combinations of the active VMs and determining these VMs' resource requirements. Each subset of VMs may contain one or more VMs; in addition the empty set (i.e., the set of no VMs) may be included to guarantee that there is at least one subset of VMs does not exceed the power available to the host platform.

Next, at processing box 316, the VMM selects the subset that has the most value to the user from the subsets of VMs identified at processing block 314 using a policy pertaining to user preferences with respect to the VMs. In one embodiment, the policy pertaining to user preferences is predetermined (e.g., defined by the computer manufacturer). Alternatively, the user is provided with an opportunity to specify his or her desired policy regarding the VMs. For instance, the user may specify the desired policy in advance (i.e., statically) or at the time the most valuable subset of VMs is being selected (i.e., dynamically). In one embodiment, the user's desired policy regarding a particular VM is maintained by an application (e.g., a resource management application) running in this VM. The application can then communicate this policy to the VMM at any appropriate point of time.

Further, all the active VMs other than the VMs selected at processing block 316 are saved and stopped. That is, the states of these VMs are saved (processing block 318) and the VMs are stopped to free the resources allocated to these VMs (processing block 320).

Subsequently, when any of the VMs that were stopped becomes active, the VMM restores the state of this VM using the saved state information. Accordingly, the VMM is able to balance between resource requirements of multiple VMs and available resources of the host platform.

Figure 4 is a flow diagram of a method 400 for assisting a guest OS that is not aware of power-management, according to one embodiment of the present invention. As described above, some guest OSs may not be equipped to handle power-manageable signals sent by the host platform. These guest OSs are referred to herein as non-power-management-aware guest OSs. In one embodiment, the VMM assists such guest OSs by intercepting the power-management signals sent by the host platform and preserving the state of the corresponding VM when necessary. Method 400 illustrates this embodiment of the present invention using an exemplary scenario of low battery. It should be noted that method 400 should not be limited to this particular scenario and may be used to assist the guest OS in various other situations without loss of generality.

Method 400 begins with intercepting a power-management signal sent to the VM that runs a non-power-management-aware guest OS (processing block 404). At decision box 406, a determination is made as to whether this signal indicates that the battery used for the host platform is low. If the determination is negative, the VMM takes no actions, and method 400 ends.

Alternatively, if the determination is positive, the VMM saves the state information of the VM (processing block 408) and powers down the host platform (processing block 410). Subsequently, when the host platform is

powered up, the VMM restores the state of the VM using the saved state information. Accordingly, the VMM prevents the non-power-management-aware guest OS from losing data during the power-down state of the host platform.

Figure 5 is a block diagram of one embodiment of a processing system. Processing system 500 includes processor 520 and memory 530. Processor 520 can be any type of processor capable of executing software, such as a microprocessor, digital signal processor, microcontroller, or the like. Processing system 500 can be a personal computer (PC), mainframe, handheld device, portable computer, set-top box, or any other system that includes software.

Memory 530 can be a hard disk, a floppy disk, random access memory (RAM), read only memory (ROM), flash memory, or any other type of machine medium readable by processor 520. Memory 530 can store instructions for performing the execution of the various method embodiments of the present invention such as methods 200, 300 and 400 (**Figures 2-4**).

It is to be understood that the above description is intended to be illustrative, and not restrictive. Many other embodiments will be apparent to those of skill in the art upon reading and understanding the above description. The scope of the invention should, therefore, be determined with reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.